ネットワークプログラミング演習環境の構築について

三　谷　和　史

はじめに

筆者の属する本学社会情報学科・社会と情報講座では，情報処理基礎，情報処理，情報科学，コンピュータネットワーク論といった科目で，C言語等を用いた情報処理・プログラミング，ネットワーク教育を行っている。これらの教育では，実習や演習といった実際にコンピュータを用いた学習が重要であり，本学では情報処理センターの実習環境を用いてこれらを行っている。基本的なプログラミング言語の演習等にはマイクロソフトのWindows環境でのVisual Studioを用いているが，ネットワークプログラミングに関しては共用のUNIX環境を用いての演習を行ってきている。

平成28年３月の情報処理センターの機種更新において，これまで用いてきた共用のUNIX環境が提供されなくなったため，今回ネットワークプログラミングに関する演習の環境を，本学の実情に合わせてどのように提供するかを検討し実現・実施したので，それについて報告する。

環境構築に至る経緯

長年に亘り情報処理センターのUNIX環境を用いて演習を行ってきたが，今回新に演習環境を構築するに至った経緯について簡単に説明する。

国立大学等では，昭和30年代半ばから学内に計算センターや計算機センターが設けられ，学術計算，当時は主に数値計算を用いた研究のための計算資源を

提供してきた。これが，昭和40年代半ばに旧七帝大に大型計算機センターが全国共同利用施設として設置され，その後，情報処理教育のための情報処理教育センターがこれら七大学の他５大学[1)]に設置された。又，その他の大学の計算センター等は昭和の終わりから平成にかけて情報処理センターとして改組されてきた。その後，これら３種のセンターの多くは再改組されて各大学で様々な名称がついている。本学の情報処理センターもこのような経緯を持ち，本学の研究のための計算資源の提供がそもそもの目的であった。

しかし，時代の変化に応じて紆余曲折を経て，現在では情報処理実習のためのPCと学内ネットワークの整備，インターネットとの接続とネットワークサービスの提供を主たる業務としており，本学での研究のための計算資源を提供する機器は近年は機種更新の度に縮小されて，今回の機種更新においては措置されなくなった。

とはいえ，この判断は正しいと考えて良いと思われる。その理由は以下の通りである。

近年の本学での研究のための計算資源の提供はUNIXを搭載したワークステーションによって行われてきたが，PCの進化によって性能差(CPU，メモリ，ディスク）が昔ほど大きくなくなってきた点が１つある。研究のための計算を自身のPCで行うことで十分賄えるようになれば，共用の設備を使う必要がなくなる。

また，ワークステーションでしか動作しなかったソフトウェアの多くが PC でも動作するようになり，その面でも必要がなくなった。

逆に，PCでは賄えないような大規模な計算や高速な計算を必要とする場合は，本学で提供されていた程度の計算資源では全くの不足であって，スーパーコンピュータやクラウドを用いた大規模並列計算を行う必要があり，それらは本学のネットワークを通じて十分利用可能である（ネットワークの帯域を除いて)。

1）室工大，名工大，九工大，後に和歌山大と広島大

そのため，定常的に研究のために本学情報処理センターのワークステーションを利用する人数が減少したため，今回不要とされたのである。

とはいえ，一時的な教育のための利用方法では十分にその用を満たしてきたので，無くなってしまうと演習が行えなくなる。

このような共用の計算機環境を使用した演習の利点は，計算機環境の保守が情報処理センターの手で行われるため，必要なセキュリティパッチ等が定期的に当てられて，学生のアカウント管理も行われる点である。また，学外から講義時間外に利用することも可能である。

問題点は，学生が作ったバグのあるプログラムが研究用のプログラムの実行と同時に行われるため，無駄に動き続けて計算資源を消費する点である。さらに，意図的にもしくは意図せずに，fork爆弾と呼ばれる再帰的に多数のプロセスを生成させてOSのプロセステーブルを埋め尽くして，システムのリブート以外の対処法のない状況を生じさせ，システムの正常利用を妨げさせうる点もある。bashでは:(){:|:&};:という13文字がその例となる。

その他，ネットワークプログラムの実行がサーバとクライアント共に1台の計算機となるため，指定するホストがlocalhostのみとなって実際にネットワークを使っている実感が多少薄れるという点もある。

環境をどう構築するか

次に，演習環境をどのように構築するかを考える。1つの方法はこれまでと同様なUNIX環境のワークステーション等を用意するという方法である。当研究室でそのような環境を用意はできないのでこの方法は無理である。また，学内のサービス用のサーバ上に仮想マシンを作成して，これまでと同様なUNIX環境を構築するという方法もあるが，一時的な教育環境のために学内の資源を割くのは許されない。

学生の実習に使えるのは，やはり実習室に置かれているPCとなるのが適当である。

現在の実習室のPCは，4GBメモリに128GB SSDでWindows VHD-bootを用いたシンクライアント方式で運用されており，現在使われているwindows 10のイメージの他にUNIX系のOSのイメージを作成して，そのイメージでPCを起動するという手法は可能である。しかし，PC側にイメージをキャッシュするためのSSDの容量が少ないので，この手法は難しかった。

UNIX系のOSではlive imageと呼ばれる，起動するとハードディスクにインストールすることなくそのまま使えるものがあり，USBメモリやDVDにこれを書き込んでPCをこれからブートして使用するという方式も考えられる。しかし，これは開発に必要な環境を後から追加したり，作成したプログラムを保存する手段を別に持たなくてはならなく，さらにPCに接続された物理デバイスを自由にすることが可能であるため，既存のSSDの中身を壊すことも可能であり，この方式を用いるのは今回は適当ではないと考える。

その他，クラウド環境を購入して行うという手法も考えられるが，費用の面で今回は見送らざるを得ない。

そこで，今回はPC上に仮想計算機を構築して，そこで実習を行うという方式を選択する。本学のPC環境であるwindows 10で動作する仮想計算機の方法としては，フリーで使えるOracle製のVirtualBox，MicrosoftのHyper-V，VMware社の製品等がある。演習が終わった後，学生が自身のPC環境に仮想計算機のイメージを移動して作業を続ける場合を想定すると，PCの環境がwindows以外のMac OSXでも動作する点でVirtualBoxかVMwareを考え，本学のPC上にセンターとしてインストールして保守していく点を考えると有償であっても製品のほうが安心できるので，今回の機種更新時に仮想計算機のイメージ作成用にVMware Workstation 12 for Linux/Win ESDを購入して頂き，出来上がった仮想計算機のイメージを実行するためには，無償で利用できるAcademic VMware Workstation 12 Player12.0.5を PCにインストールして頂いた。

当初は，仮想計算機のイメージはファイルサーバ上の学生のホームディレクトリに置かれ，アクセスはネットワーク経由となるが，PCは1Gbpsのネット

ワークでファイルサーバと接続されているので，速度的に耐えられないことはないであろうという目論見であった。現実には実習を行う本学の第4実習室が100Mbpsの接続であったため，速度的に耐えられるか実験を行い，無理であればUSBメモリ上にイメージを置いて使うことを考え，結果USBメモリ上にイメージを置いて行うこととなった。

OSの選択と資源の配分

実習環境を学生用PC上の仮想計算機であるVMwareとすることに決定したので，次はそこで動作させるオペレーティングシステム（OS）の選択を行う必要がある。これまで同様に，UNIX系のOSを使用することを考える。

UNIXの誕生から今日に至る歴史・進化については［1］や［2］，［3］に詳しい。AT&Tのベル研究所のUNIXのコードから派生した，もしくは仕様の振舞いを真似たシステムは多数存在する。そして，現在UNIXという名称は，オープン・グループ[2)]が商標を所有しており，その管理によるSingle UNIX Specification[3)]を満たすシステムのみがUNIXを名乗ることができる。これらのほとんどは，SYSTEM Vのコードをベースとした商用のものである。

そこで，正式にUNIXを名乗れないシステムをUnix系システムと呼ぶことが多いが，ここではゼロックスが複写機の一般名称として用いられるのと同様に，UNIXという名称を普通名詞化したものと見做して特に区別せずに用いている。

まず初めに，現在のUNIXに至る歴史について簡単に述べる。

2）http://www.opengroup.org/
3）http://www.unix.org/what_is_unix/single_unix_specification.html

ベル研究所のMulticsからの撤退

MITのProject MACの一環として，MITのCTSS（Compatible Time Sharing System）の後継となるOSの開発が行われた。それが1965年からMITのFernando José Corbató等を中心にベル研究所，GE（General Electronic）によって始まった，独創的で先進的なTSS OSであるMultics（**Mult**iplexed **I**nformation and **C**omputing **S**ervice）である。しかし，この開発はPL/Iコンパイラの開発の遅れにより頓挫しそうになったため，1969年にベル研究所は見切りをつけてプロジェクトから撤退した。ベル研究所からはPeter Gabriel Neumann（Peter Neumann），Malcolm Douglas McIlroy（Douglas McIlroy），Robert Morris（Bob Morris），David J.“Dave”Farber（Dave Farber），James F. Gimpel（Jim Gimpel），Joseph F. Ossanna（Joe Ossanna），Stuart Feldman（Stu Feldman），Kenneth Lane Thompson（Ken Thompson），Rudd Canaday，Dennis MacAlistair Ritchie（Dennis Ritchie），Brian Wilson Kernighan（Brian Kernighan）が参加していた。

尚，その後もMulticsはMITとGEで開発が続き，1970年にGEのコンピュータ部門がHoneywellに Multicsもろとも売却されて，結果としてHoneywellが商品化した。1985年頃までGEの設計に基づいてシステム開発は続き，80カ所程度の大学・研究所・政府機関で使われた。1991年にはフランスのBull社がHoneywellのコンピュータ部門を買収している。最後のシステムは2000年まで稼働していたが，Multicsは商業的に成功したとは言えない。

現在Multicsに関する情報は公開されており[4)]，Multicsのソースとドキュメントも公開されている[5)]。更には，当時の計算機DPS8Mのエミュレータ上でMulticsを動作できる環境も公開されている[6)]。

4）http://multicians.org/index.html
5）http://web.mit.edu/multics-history/
6）http://swenson.org/multics_wiki/index.php?title=Main_Page

UNIX at Bell Labs

1969年，Multicsから撤退したベル研究所で，Multicsに携わっていたKen Thompsonが研究所の物置にあったDEC[7]のミニコンPDP-7に，Multicsから機能を削ぎ落としたUnics（UNiplexed Information and Computing Service）をアセンブラで作成し，それが後にUNIXと改称された。これが所謂Version 0のUNIXである。その後，これはDennis Ritchie, Brian Kernighan, Douglas McIlroy, Michael E. Lesk（Mike Lesk），Joe Ossanna等の助けを受け DEC PDP-11/20へ移植された。PDP-11は16ビットのミニコンである。

1973年にはDennis RitchieがUNIXのカーネル部分をＣ言語で書き直し，移植性が増してPDP-11の複数機種で用いられるようになった。

UNIXのバージョンは公式マニュアルのEditionで一般的に表される。正確にはUNIX Time-Sharing System, Nth Edition（Version N）といった名称となるが，これをVNと略記することとする。順に述べると，所謂V0は1969年にPDP-7で動作した。UNIXと改称されたV1は1971年11月にPDP-11/20で，V2は1972年６月にPDP-11/20で，V3は1973年２月にPDP-11/45で動作している。ここまでは全てアセンブラで作成されている。その後，Ｃ言語で書き直され，V4は1973年11月にPDP-11/45で動作した。ベル研究所の外に持ち出されたのはこのV4からと見られている。V5は1974年６月にPDP-11/45,40で動作し，V6は1975年５月にPDP-11/40,70で動作した。そしてV7が1979年１月にPDP-11/45,70, Interdata 8/32で動作した[8]。また1979年６月には，V7をDECの32ビットコンピュータであるVAX-11に移植したUNIX/32V Time-Sharing System Version 1.0（32V）も動作した。32Vも含めてこれらはプロセスのスワッピン

7）Digital Equipment Corporation. 後にCompaq Computer Corporationに買収され，CompaqもHewlett-Packard Companyに買収された。

8）Interdataは1966年から16bit及び32bitのミニコンの開発を開始し，1973年にPerkin-Elmerに買収されているが，1974年に発表したInterdata 7/32は最初の32bitコンピュータの一つでIBM System/360のアーキテクチャをベースにしていた。

グのみで仮想記憶の機構を有していなかった。

1973年10月にFourth ACM Symposium on Operating Systems Principles[9]でUNIXについて部外に初めて発表がなされた。これが後にCACMで公表されている［4］。このときUCB（University of California, Berkeley）のBob Fabryが講演を聴いて，UNIX V4のコピーを1974年１月にバークレーに持ち帰ったのが，後のBSDの始まりとなる。The Bell System Technical Journalでは２度UNIXの特集号が組まれた［7］，［8］。1975-76年にかけてKen Thompsonがサバティカルで母校のUCBに１年間客員教授として滞在した当時，V6がUCBにもたらされた。同時期に，Chuck Haleyと William Nelson Joy（Bill Joy）が大学院生でとしてUCBに入学。２人はKen Thompsonが手掛けていたPascalシステムに興味を示しこのシステムの改良を始める。UCBで彼の教えを受けた２人やEric Emerson Schmidtといった1954，55年生まれの学生がBSDの開発の力となった。

ベル研究所は元々 WE（Western Electric）の研究部門とAT&Tの技術部門が母体であり，両者が半分ずつ出資して設立された。この当時は，1956年の独占禁止法違反の訴訟での和解判決合意により，ベル研究所の親会社のAT&Tはコンピュータ産業への進出を禁止されていたため，V4の時代にKen ThompsonはUNIXをソースコードと共に希望者に無料で発送しはじめた。最初のテープはコロンビア大学であった。当時は定まったライセンスすら無かった，V5の時代から正式に大学に対して教育目的にメディアとマニュアルのコピー代程度でライセンスが行われ，幾つもの大学でOSの講義に使われた。V6からはWEが配布業務を担当し，企業や政府機関もライセンスの対象となった。これら初期のベル研究所，WEによるUNIXライセンスによる配布はテープかRK05のディスクパックで行われ，“No support, no trial period, no warranties, no advertising, no bug fixes, and payment in advance.”がベル研究所の姿勢であった。

そのため，V4時代からUNIXユーザによる会合が開かれ情報交換が行われて

9）http://dblp.uni-trier.de/db/conf/sosp/sosp73.html

いた。最初の会合は1974年５月15日で，CACMに論文が発表される２ヶ月前である。その後も会合は開かれ，ユーザ会はUnix Users Groupといい，UNIX NEWSというニュースレターを刊行した。1975年６月20日が最初の号[10]である。これは1977年のMay-June号まで続き，July 1977より*;login:* という名称になった。これは，AT&Tの法務部よりUNIXの名称を使うなとのクレームによるものである。会はUSENIX[11]となった。

UCB以外にも，例えば本学の提携校であるオーストラリアのウーロゴン大学ではUNIXを入手して，Richard Millerがベル研に先立ち1976年からInterdata 7/32へV6及びV7の移植を行っている[12]。

また，オーストラリアではJohn Lions等が立ち上げたAUUG（Australian Unix systems User Group）[13]というUSENIXと並んで古いユーザグループがあり会報も出していたが，現在は活動を停止している。ちなみに，日本ではjus（日本UNIXユーザ会）[14]が1983年６月より現在も活動している。

AT&TはV7の頃からUNIXの製品化を考え始め，この頃からライセンスが厳しくなった。

そのため，Lions book［5］と呼ばれたV6のカーネルのソースコードを解説した当時唯一の本は，V6のライセンス下では授業で使うことができたが，V7ではそれができなくなった。以降は違法コピーという形で広く流通したが，現在は再出版されている。

更には，授業で自由にソースコードが使えなくなったため，アムステルダム自由大学（Vrije Universiteit Amsterdam）のアンドリュー・S・タネンバウム（Andrew Stuart“Andy”Tanenbaum）は，OSの教材用にV7の互換シス

10）https://www.usenix.org/system/files/login/articles/login_apr15_16_unix_news.pdf
11）https://www.usenix.org/
12）http://www.uow.edu.au/content/groups/public/@web/@inf/@scsse/documents/doc/uow103747.pdf
13）https://en.wikipedia.org/wiki/AUUG
14）https://www.jus.or.jp/

テムを再設計し，1987年に著書のOperating Systems: Design and Implementation［6］で取り上げた。機能上新しさはないが，マイクロカーネル構造を採用するなど洗練され，ソースコードの行数は現在のMinix3でもマイクロカーネル本体が4000行弱となっている。

その後，AT&TはV7に社内で使われていた幾つかのUNIXのバージョン（PWB/UNIX[15]，CB UNIX[16]，UNIX/RT[17]）や32Vの機能をつぎはぎして，1982年にUNIX System IIIを社内のUNIXサポート組織であるUSG（Unix Support Group）からリリースした。結果，ベル研究所はUNIXを配布する権限を奪われた。また，これまでのバージョンを含めて社外への配布業務を担当していたのは WEであったが，System IIIがWEの配布する最後のバージョンとなった。System IIIは，HP-UX，IRIX，PC-UX，IS/3，Venix，Xenixなど多数のサードパーティ製品を生んだ。特にPC-UXはNECのPC9801でも動作し［9］，筆者も北大時代に使用した経験がある。

これは，1982年にAT&Tの解体で独占禁止法違反の訴訟が決着し，1984年1/1にAT&Tが分割されることとなり，AT&Tは通信業務以外の分野への参入が認められUNIXを製品化できるようになったためである。

System IIIという名称は，ベル研究所内部で使われていたUNIX/TS 3.0.1およびCB UNIX 3の外部リリースに相当するためで，System I, IIの名称を持つUNIXは存在しない。UNIX/TS 4.0は公式リリースされなかったためSystem IVも存在せず，次のバージョンはUNIX/TS 5.0に基づいたSystem Vとなった。USGはUNIX System V R1（’83），R2（’84），R3（’86），R3.2（’88）と製品をリリースしていった。さらに，R2の解説を行った［10］が出版され，その普及に一役買った。1989年にはUSL（UNIX Systems Laboratories）が

15）Programmer’s Workbench. V6ベースの1.0とV7ベースの2.0があった。
16）Columbus UNIX. Ohio州Columbusの研究施設によるIPC等が増強されたバージョン
17）UNIX Real-Time。最初はMERT（Multi-Environment Real-Time）と呼ばれたtime-sharing OSとreal-time OSのハイブリッド。

UNIXの開発とライセンス業務を目的として正式に設立されるが，それ以前からあったUSLは Sun Microsystemsと共同でR3と4.3BSD，SCO社のXENIX，SunOSの技術を統合したSystem V R4を1988年に発表した。R4は多くのベンダーに採用された。Sunは Solaris 2（SunOS 5.0）としてR4を採用し，これまでのBSDベースから方向転換した。1994年にはR4の解説を行った［11］が出版された。その後，R4.0MP，R4.1 ES（Enhanced Security）がリリースされた。1992年，USLはNovellとジョイントベンチャー Univellを創業し，R4.2がUnivel UnixWareとしてリリースされた。USLは同年にUNIXの資産と共にNovellに売却された。1995年にはR4.2 MPがUnixWare 2としてリリースされた。

ベル研究所内部では，V7の後もResearch Unixとして V8（’85），V9（’86），V10（’89）と開発はされていったが，外部に出回ることはほとんどなかった。その後は分散OSのPlan 9 from Bell Labs（Plan9, 1st ed. ’93）やInferno（’95-）といった方向に転換し，それらは後に自由に使えるようになった。

その後のベル研究所について簡単に述べる。1996年，AT&Tはベル研究所を含めたAT&Tテクノロジーズを独立させ，ルーセント・テクノロジーズとした。2005年の 8 月にベル研究所のUNIXの生みの親の部署であるDepartment 1127が公式に解散し，UNIXを創ってきた多くの研究者が研究所から去っていった。2006年，ルーセント・テクノロジーズは仏のアルカテルとの合併に合意し，2007年, ルーセントのベル研究所とアルカテルの研究開発部門が合併し，新たなベル研究所となる。2015年，芬のノキアがアルカテル・ルーセントを買収し，2016年 1 月にアルカテル・ルーセントとベル研究所は正式にノキアの傘下となった。

BSD

先に述べたように，UCBでは1974年からV4が使われ始め，その後V5，V6が使われていた。BSD（Berkeley Software Distribution）はUCBの学生達，その後CSRG（Computer Systems Research Group）により1977年から95年にか

けて開発・配布されたソフトウェア群及びUNIX OSのことである。現在は，BSDの子孫の総称として使われることもある。ソースコードのベースはAT&TのUNIXと共通であり，BSDを使うためにはAT&Tのライセンスを保持している必要があった。

次に，BSDの歴史を簡単に述べる。1977年にBill Joyがバークレーで改良されたソフトウェアを1BSDとしてまとめ始め，1978年3/9にリリースした。V6へのアドオンという形式であり，Pascalとexが含まれた。その後1979年5月に2BSDがリリースされ，viとC Shellが含まれた。2BSDは PDP-11をターゲットとしたが，3BSDからはVAX-11をターゲットとした。2BSDの開発自体はその後も続き，2.9BSDでは4.1cBSDからのコードを含み，それまでパッチの集まりであったものがV7の修正版としての完全なOSとしてリリースされた。1992年に2.11BSDが最終リリースされたが，それ以降も2008年12/31にパッチ447が，2010年1/5にパッチ448がリリースされるなど，ボランティアで保守が続いている[18)]。

1979年，UCBのVAX-11に32Vがインストールされたが，スワッピング機能のみで仮想記憶がなかったため，バークレーの学生達が仮想記憶を実装し，2BSDのユーティリティーを移植して，完全なOSとして3BSDが1979年末にリリースされた。

DARPA（Defense Advanced Research Projects Agency：国防高等研究計画局）はVLSI Projectのための標準UNIXプラットフォームとして3BSDに注目し，UCBのBob Fabryに資金提供を決めた。この資金を得て1980年にCSRGが組織された。

CSRGは3BSDを改良して4BSDを1980年11月にリリースした。これはVAX-11/750の出荷と時期を同じくしている。1981年6月には4BSDのカーネルのチューニングをBill Joyが行った4.1BSDがリリースされた。当時5BSDと呼ぶ予定が，UNIX System Vとの混同を唱えるAT&Tからの異議により4.1BSDとな

18）https://github.com/larsbrinkhoff/2bsd/tree/master/2.11-patches

り，以後はマイナーバージョンアップの形となった。その後，4.2BSDが1983年8月にリリース，4.3BSDが 1986年6月にリリースされた。CSRGは古臭くなったVAXから新しいプラットフォームへの移行を始め，何台かの実機を提供してくれたCCI（Computer Consoles Inc.）の68kベースのPower 6/32（コード名“Tahoe”）への移植が行われた。結局CCIが戦略を変更して，このアーキテクチャを中止したため短命に終わったが，1988年6月の4.3BSD-Tahoeという移植版は機種依存コードと機種共通コードの分離をもたらし，将来の移植性を向上させた。4.3BSD-Renoが1990年初めにリリースされたが，これは4.4BSDの中間リリース的性格を持ち，ギャンブルの街Renoを冠した。RenoはPOSIXへの肩入れが強く不評だった面もあった。そして，4.4BSDが1996年6月にリリースされた。これはAT&Tのライセンスに抵触しない4.4BSD-Liteと従来通りのライセンスを要する4.4BSD-Encumberedの2種類がリリースされている。その後，1995年に4.4BSD-Lite Release 2がリリースされCSRGは解散した。

BSDの解説本としては，4.3BSDに対して［12］，4.4BSDに対して［13］が出版され，後にはFreeBSDを対象として［14］及び2版の［15］が出版されている。

現在，CSRGのアーカイブはCD 4枚組で頒布されている[19)]。The Unix Heritage Society[20)]には歴史的，非主流なものを含めたUNIXの資料がある。更には，SIMH[21)]という昔のコンピュータのシミュレータの上で昔のUNIXを動作させることが出来，その方法が公開されている[22)]。その中には，プリントされた紙からOCRで復元されたUNIX V1もある[23)]。

19）http://www.mckusick.com/csrg/
20）http://www.tuhs.org/
21）http://simh.trailing-edge.com/
22）例えばhttp://gunkies.org/wiki/Main Page
23）https://code.google.com/archive/p/unix-jun72/

TCP/IP on BSD

前節の説明で一箇所端折った部分がある。それは4.1BSDから4.2BSDとなる時期に導入されたTCP/IPの実装についてである。ここでは，このネットワークコードの実装の歴史について述べる。

現在のインターネットは，1969年から始まったARPANETに始まるものである。ARPANETでパケットの転送をつかさどるのはIMP（Interface Message Processor）と呼ばれるミニコン[24)]で，現在のルータの原型にあたる。ホストとIMPの間はHOST-IMPプロトコルで，IMP間はIMP-IMPプロトコルで転送される。IMP間で再送を行うため，送信元HOSTに繋がるIMPは，受信側ホストが繋がるIMPからパケットの到着を通知されてから再送用のバッファを解放し，一定時間たっても通知がなければ問い合わせの後再送する。HOSTで動作するプロトコルはNCP（Network Control Program）と呼ばれ，プロセス間接続とフロー制御を行い，HOST-IMP間とIMP-IMP間はBBN #1822プロトコルが用いられていた。

1974年にTCP/IPの原型となる論文［16］が発表された。当時はTCPと呼ばれIPと一体のものであった。IPのバージョンの変遷はRFC750［17］によれば，1977年3月にV0，1978年1月にV1，2月にV2とV3と進み9月にV4となった。この時点でTCPとIPが別のプロトコルとして分離した。1981年にTCPはRFC793［18］，IPはRFC791［19］で定義された。

DARPAはARPANETのプロトコルをTCP/IPに変更しようと計画し［20］，そのための実装環境の1つとしてBSDを選択し，CSRGに対して資金提供を行った。

ARPANETを運用しているBBN（Bolt Beranek and Newman）は初期の段階からTCPの開発を行っており，TENEX（BBNから後にDECへ売却されTOPS-20となる）への実装を担当し，他にもPDP-11上のV6上にJack Haverty

24）初期にはHoneywell DDP-516の改造型が，その後Honeywell 316等が使われた。

が実装を行っていた。1980年９月にBBNが4BSDに対するBBN VAX TCP/IPの実装契約をDARPAと結び，Rob Gurwitzが実装を担当した［21］。彼はHavertyのコードを使うこともできたが，NCPの実装から始めたHavertyのコードはTCPとIPのコードが分離されておらず，結局Gurwitzは最初から作り直した。

BBN VAX TCP/IPのプロトタイプの実装が4.1BSDにインポートされて4.1aBSDが作成されたが，Bill Joyはパフォーマンスが出ないため，独自の変更・拡張を開始した。BBNは広域のネットワークでの使用を考えて実装を行ったため，BBNのプロトタイプの実装ではLANで使用する10MbpsのEthernetでのパフォーマンスが，CPUの100％負荷時でも56KB/sec程度と低かった。Bill JoyとSam Lefflerによる変更・拡張によりEthernetで700KB/sec程度までパフォーマンスが向上した。4.1bBSDは Marshall Kirk McKusickによるBerkeley Fast File Systemが実装されたが，snap shotとしては公開されなかった。4.2BSDのリリースの数ヶ月前，1983年４月に4.1cBSDが公開された。この公開は，DARPAとの契約期間の満了に間に合わせるためのものでもあった。尚，この当時はCSRGからの4BSD/4.1BSDにはBBN VAX TCP/IPのコードは同梱されておらず，BBNから入手する必要があった。ベル研究所のResearch Unix V8はこれを元にした。

4.2BSDの仕様を決めるため，DARPAのDuane Adamsはsteering committeeを組織し，UCBからBob Fabry, Bill JoyそしてSam Leffle, BBNからAlan NemethとRob Gurwitz，ベル研究所からDennis Ritchie，StanfordからKeith Lantz，Carnegie-Mellonからは Rick Rashid，MITからは Bert Halstead，ISI[25)]からは Dan Lynch，そしてUCLAからGerald J. Popekがその任に当たった。委員会は1981年の４月から1983年の６月まで会合をもった。4.2BSDは同年８月にリリースされた。尚，前年の1982年の晩春にBill JoyはUCBを離れ，Sunに参画した。

25）Information Sciences Institute. University of Southern California（USC）に置かれた。

16番目の社員であるが，組織図上では会長の Scott G. McNealyより上，最上位に置かれていた。Joyの後を継いで4.1cからはLeffleがプロジェクトを担当した。4.2BSDのリリース後，Leffleはルーカスフィルムに移り，Mike Karelsがその後を継いだ。

4.2BSDのネットワークコードはBBNが引き渡した公式バージョンの実装ではなく，BBNのプロトタイプのコードにBill Joy等が変更・拡張を行った実装に変更されており，BBNはそれに反発して次の4.3BSDではBBNの公式実装を使うようDARPAに求めた。Karelsは双方の良いところをマージすることも考えていたが，結局どちらの実装を選ぶかをユーザ毎に決められるようにしようとした。しかし，不必要な非共通性が起こることを懸念したDARPAはどちらか一方の実装に絞ることにした。DARPAが依頼した中立な立場のBallistic Research LaboratoryのMike Muuss[26]による１ヶ月の試験の結果，BBNの公式実装よりも4.2BSDの実装が僅かだが優れているとの報告を得て，次の4.3BSDでもこの実装が引き継がれた。報告書の概要としては，4.2BSDのコードの方がBBNのコードより効率的だが，輻輳時の扱いはBBNのコードが上である。4.2BSDのコードは全てのテストを完全に通過したが，BBNのコードは一部のストレステストでパニックを起こしたというものであった。また，BBNのコードからアイディアを拝借している4.2BSDのネットワークコードの実装が，CSRG独自のものであるという判断が得られた事も収穫であった。

FreeなBSDに向けて

BSDのコードはAT&Tのライセンスを保持している組織でなければ入手できなかった。また，ライセンスの値段も高くなった。そのため，AT&Tプロプライエタリーなコードを除去し，AT&Tのライセンスに縛られない形での配布をCSRGは模索した。4.3BSD-Tahoeから1989年６月にNet/1が，Renoか

26）TCP/IP Digestのモデレータ。pingやttcpの作者。この試験のためttcpを作成。

ら1991年6月にNet/2がAT&Tのライセンスに縛られない独自のソースコードとしてリリースされた。Net/1はBSDの手によるコマンドのソースやOSの一部分のソースコードであったが，Net/2ではAT&Tのコード由来のほとんどが書き直されて，ほぼ完全なOSのソースコードとして公開された。しかし，それだけではOSとして動作することは出来なかった。ライセンスにより公開できなかったコードは，関数の内部が“*Body deleted.*”とマークされており，kern_acct.c, kern_exec.c, kern_physio.c, subr_rmap.c, sys_process.c, tty_subr.c, vfs_bio.cの7つのファイルが該当した。このNet/2はIntel 80386系への移植の基盤となった。

そこでNet/2で足りないファイルを，William Frederick（Bill）JolitzとLynne Greer Jolitzの夫妻が作成して，386BSDとして公開した。1992年2月に386BSD 0.0が，1992年7月にはbug fix版の386BSD 0.1がリリースされた。その後は開発が滞り，寄せられた膨大なパッチからユーザらは独自にUnofficial 386BSD Patchkitを製作した。しかし，その後もbug fixや新たな開発は行われなかった。386BSD自体は短命に終わったが，それをベースにNetBSD[27]とFreeBSD[28]プロジェクトが開始された。最終的には1994年に386BSD1.0がJolizからリリースはされたが[29]，それで終わりとなった。

その他には，BSDi（Berkeley Software Design, Inc.）が，1992年にBSD/386（後にBSD/OSに改称）を作成して販売した。社員の殆どはCSRGのメンバーであった。NDA（Non-disclosure agreement）を結んでの周辺デバイスの開発が可能なため，対応周辺デバイスが充実しており，性能面でも高い評価があった。しかし，BSDiはUSLにより1992年に訴訟をおこされ，これが決着するまでの2年間，Net/2の配布は差し止められた。

尚，BSDiは2000年にFreeBSDのCD-ROMを販売していたWalnut Creek CDROMを買収したが，2001年にはBSDiの一部（BSD/OSを含む）を組込OS

27）http://www.netbsd.org/
28）http://www.freebsd.org/
29）http://gunkies.org/wiki/386BSD_1.0

を主たる業務とするWind River Systemsに買収された。Wind River Systemsはその後もBSD/OSの開発・販売を続けていたが，2003年，BSD/OS 5.0のリリースをもってBSD/OSの権利を保持したまま開発・販売を終了してLinuxに方針転換し，本業のVxWorksとの２本立てで営業している。

CSRGの時代からFreeとなったBSDの歴史は，SVN repositoryとして公開されている[30)]。

BSDの子孫たち

386BSDの意志を継ぐ形でNetBSDとFreeBSDが始まった。

NetBSDは386BSDのユーザが集まってシステムの維持と今後の協力を目的に，CSRGと似た研究中心の開発姿勢を維持しながら，多くのプラットホームをサポートしていくことを目的に立ち上げた[31)]。NetBSDのディストリビューションはインターネット経由のみで配布され，発足当時から技術に長けたパワーユーザを対象としている。

386BSDのユーザが進めようとした，「パッチキットを適用した状態の386BSDのclean-up snapshot製作プロジェクト」をJolitzが拒否した。そこで，パッチキットの最後の取りまとめ役であったNate Williams, Rod Grimes, Jordan Hubbard等が自分達で新しいOSの開発を行う事を決意し，1993年，NetBSDの発足から数ヶ月後に，The power to serveを掲げてFreeBSDプロジェクトを開始した[32)]。

彼らはPCを専門にサポートし，技術的にそれほど詳しくないユーザを対象にすることを目的にした。そして彼らはWalnut Creek CDROM社と協力して，インストール作業を簡易化するスクリプトをCD-ROMに付けてFreeBSDソフトウェアとして配布した。CD-ROMのおかげでネットワークに接続できなく

30）https://svnweb.freebsd.org/csrg/
31）http://www.netbsd.org/about/history.html
32）https://www.freebsd.org/doc/handbook/history.html

てもインストールが可能となった。さらに，Linuxのバイナリプログラムを動かせるようにemulation modeを追加した。これでFreeBSDのユーザは急増した。FreeBSD 1.1.5.1まではNet/2がベースであり，Net/2が公開停止になってからは4.4BSD-Liteをベースとして FreeBSD 2.0が開発された。

筆者はNetBSD 0.9とFreeBSD 1.1.5の時代からのユーザである。

OpenBSD[33]やDragonFly BSD[34]は開発方針の違いや開発グループの諍いによってそれぞれNetBSD，FreeBSDからフォークした。これら４つのOSを簡単に纏めたのが表１である。

尚，表１中の多数１[35]，多数２[36]は脚注参照のこと。現リリースコンパイラはamd64について調べたもので，アーキテクチャによっては異なるかもしれない。

表１：BSD系OSの比較

OS	FreeBSD	NetBSD	OpenBSD	DragonFly BSD
最初のリリース	1993/11/1（v1.0）	1993/4/20（v0.8）	1996/10/1（v1.2）	2004/7/12（v1.0）
最新リリース版	10.3	7.0	6.0	4.6
創始者	Nate Williams Rod Grimes Jordan K. Hubbard	Chris Demetriou Theo de Raadt Adam Glass Charles Hannum	Theo de Raadt	Matthew Dillon
由来	Net/2, 4.4BSD-Lite	Net/2, 386BSD	NetBSD 1.0	FreeBSD 4.8
カーネルタイプ	モノリシック ＋モジュール	モノリシック	モノリシック	ハイブリッド
対応アーキテクチャ	amd64,arm,arm64, i386,mips,pc98, powerpc,riscv, sparc64,x86	多数１	多数２	x86-64
パッケージ管理	pkgng（ports）	pkgsrc	OpenBSD package tools, ports	DPorts, pkgng
現リリースコンパイラ	Clang 3.4.1	GCC 5.3.0	GCC 4.2.1	GCC 5.3.1

33）http://www.openbsd.org/
34）https://www.dragonflybsd.org/
35）http://www.jp.netbsd.org/ja/ports/によると57ポート。“Of course it runs NetBSD”
36）http://www.openbsd.org/plat.html参照

各々の特徴をもう少し詳しく述べる。FreeBSDはIntelのx86を対象に開発が始まった。Linuxと異なりカーネルとユーザランドの両方を含めて1つのOSであり，OS側にはGPLのものを含まないようにしている。OpenSolaris由来のZFSをFreeBSD 9.0からサポートしている。信頼性や安定性と性能と速度のバランスを重視しつつ，最新のPC，デバイス，ソフトウェアへの対応や先進的な機能の採用も可能な限り目指している。

NetBSDは様々なアーキテクチャへの対応を至上命題として開発されており，信頼性や安定性は副次的に重視している。速度や性能については可能な範囲で目指している。コンパイラ，アセンブラ，リンカ，その他のクロスコンパイルに完全対応したツールチェーン一式を持ち，どのアーキテクチャでも他のアーキテクチャの開発ができる。

OpenBSDはNetBSD 1.0からフォークして信頼性とセキュリティを至上命題に開発が進められた。最新のPC，デバイス，ソフトウェアへの対応や先進的な機能の採用，使いやすさ等は一切無視する。通常のインストールではほとんどのサービスが起動せず，何もできないと揶揄されることもあるが，デフォルトインストールでのリモートエクスプロイトセキュリティホールが過去2つしか発見されていないことを売り文句としている。また，OpenSSH等のOpenBSDのコンポーネントのためのプロジェクトは多彩で成功している。GPL排除に熱心なOpenBSDはコンパイラをgccから変更しようとpccの使用を検討したが結局止め，Clangへの変更を模索している。

DragonFly BSDは，SMPへの対応をFreeBSDとは異なった形で行うためにFreeBSD 4.8からフォークした。その後の開発方針はFreeBSD 5.xとも全く異なり，モノリシックカーネルとマイクロカーネルの両方の性質を併せ持つハイブリッドカーネルとなり，AmigaOSに触発された軽量カーネルスレッド（LWKT）や軽量メッセージシステムが実装され，ファイルシステムも冗長過ぎるZFSをMatthew Dillonがシンプルに再設計したとも言える，独自のHAMMER及び HAMMER2が開発されている。

その他としては，FreeBSDをデスクトップ環境の整備に特化したPC-

BSD[37]がある。ベースはFreeBSDで，それをデスクトップですぐ簡単に使えるよう，KDEが標準GUIとして用意され，インストールもGUIを使った独自のものである。2005年からリリースが始まり，現在の最新版はFreeBSD 10.3をベースとしたPC-BSD 10.3である。2016年9月1日にPC-BSDはTrueOS[38]に進化した。

同様なFreeBSDのデスクトップ環境としては，GhostBSD[39]がある。以前はGNOMEが，バージョン3.5からはMATEが標準GUIとなっている。2010年からリリースが始まり，現在の最新版はGhostBSD 10.3である。

そして，2012年からArch Linuxのシンプルで簡素なディストリビューションの理念やpacmanというパッケージ管理システムを利用しているArch BSDが，2015年5月にPacBSD[40]と改名して本年7月21日にamd64を対象としたリリースを出している。これはFreeBSD 11.0-BETA1をベースシステムとしている。

その他，2006年にFreeBSD 6.1-BETAからforkしたMidnightBSD[41]は，デスクトップ環境にNeXTSTEPの後継であるGNUstepを使ったもので，ライセンスがソフトウェアによって異なっている。最新版はMidnightBSD 0.8で，x86とamd64を対象としており，このバージョンでGCC4.2からClang/LLVM[42] 3.3へコンパイラが変更された。

更に，2014年には，ASLR（Address Space Layout Randomization）のFreeBSDへの採用を拒否されたOliver PinterとShawn Webbが，セキュリティの強化を目的にFreeBSDからフォークしたHardenedBSD[43]プロジェクトを開始した。amd64を対象アーキテクチャとして，現在HardenedBSD-10-STABLE-v46.7

37）http://www.pcbsd.org/
38）https://www.trueos.org/
39）http://www.ghostbsd.org/
40）https://pacbsd.org/
41）http://www.midnightbsd.org/
42）http://llvm.org/及び http://clang.llvm.org/
43）https://hardenedbsd.org/

が安定版である。

2015年8月には，FreeBSDの開発者でiXsystems[44]のCTOでもあるJordan HubbardやKip MarcyがFreeBSD-CURRENTからforkしたscience projectとしてNextBSD[45]を立ち上げた。これはFreeBSDのカーネルをMach IPCに対応させ，AppleやDarwinのカーネルで使われるlaunchd, notifyd, asldそしてlibdispatch等を動作させて，Mac OS Xの管理機能を統合するのが目的である。Hubbardは12年間Appleに勤めていたのでOS Xの技術の影響が強いとみられる。NextBSDのソースコードはgithub[46]で公開されている。NextBSDにインストーラやISOは無く，FreeBSD 10.x, 11.x, 12.xのフレッシュコピーの上でシステムをgit cloneしてビルドし入れ替える。

FreeBSD 10.0から新たなセキュリティ機能として，Robert N.M. Watson氏のCapsicumというケーパビリティに基づくリソース保護モデルが実装された。ケーパビリティは，従来rootの持つ特権と一般ユーザという2種類の特権レベルを，もっと細かい粒度でプロセスに与える方式で，セキュリティホールの発生を抑える効果を持つと考えられる。

ケーパビリティの機能をアドレス空間のメモリ保護に適応する取り組みとして，CHERI（Capability Hardware Enhanced RISC Instructions）［29］を導入するのがFreeBSD 9.0からforkしたCheriBSD[47]である。これはFreeBSD/MIPSをベースとし，研究開発用にプロセッサレベルからの開発が行われている。従来の仮想メモリモデルを保ったまま，コンパイラレベルでのケーパビリティの自動利用を目指している。

また，FreeBSDのディストリビューションとしては，firewallに特化したpfsense[48]やNAS（Network Attached Storage）と呼ばれるファイルサーバの

44）https://www.ixsystems.com/
45）http://www.nextbsd.org/
46）https://github.com/NextBSD/NextBSD
47）https://github.com/CTSRD-CHERI/cheribsd
48）https://www.pfsense.org/

FreeNAS[49]がある。

Bitrig[50]はOpenBSD5.6から2012年にフォークしたプロジェクトで，安全性維持のために保守的なOpenBSDより制限を緩め，コンパイラもGCCからClang/LLVMに変更し，i386とamd64以外にARMのサポートも開発が始まった。OSのベース部分の脱GNUを進めている。

少し古いが，2005年度のBSD系OSの利用状況の調査結果[51]によると，複数回答での調査で，FreeBSDが77％，OpenBSDが32.8％，NetBSDが16.3％，DragonFly BSDが2.5％，その他が6.6％で，FreeBSDの利用が多いことが判明した。

Linuxについて

タネンバウムのMINIXに刺激を受けて，また彼がMINIXを学習用のOSから実用的なOSに発展させる気が無いことに反発して，Linus Benedict Torvaldsが1991年に公開したのがLinuxの始まりである。MINIXがV7の互換を目指して始まったので，Linuxも同様にV7やSystem IIIの互換から始まった。ソースコードとしてはAT&Tとは全く無関係な独自なものである。ネットワークスタックのコードも独自である。Linux自体がLinux is not UNIXというrecursive acronymでもある。

現在のLinuxは，異なるOS実装に共通のAPIを定め，移植性の高いアプリケーションソフトウェアの開発を容易にすることを目的としてIEEEが策定したアプリケーションインタフェース規格であるPOSIX.1（IEEE Std 1003.1）[52]に準拠したOSである。

1992年にはニュースグループcomp.os.minix上で，タネンバウムがLinuxを

49）https://ja.wikipedia.org/wiki/FreeNAS
50）https://www.bitrig.org/
51）https://ja.wikipedia.org/wiki/BSD
52）http://standards.ieee.org/develop/wg/POSIX.html

1992年の時点でモノリシックカーネルを採用するのは時代遅れだと批判し論争となった。今の時点で考えると確かにMINIXの採用したマイクロカーネルが優れていると思われるが，時代遅れの枯れたモノリシックカーネルであることが逆に開発に参加できる敷居を下げ，広まる要因の１つであったとも思われる。

Linuxはカーネル以外に，カーネルに様々なソフトウェアを同梱したディストリビューションが多数ある[53)]。

2003年から始まるNovell・SCO裁判ではLinuxも巻き込まれているが，2010年にはUNIXの著作権はNovell側にあるということで一応の決着を見た[54)]。OSS（Open Source Software）では常にライセンス関係のトラブルが懸念される。

Linuxは中心となるカーネル以外はGNU[55)]のプロダクトを多く使っており，GNUの公式OS開発プロジェクトのHurdの完成が遅れる中，GNU側はLinuxをGNU/Linux OSと呼んでいる。

1983年から始まるGNU Projectの目的は自由なるOSを作ることで，それが元来はHurdである。当初はGNU Emacsに始まりOSを開発するための環境の整備が行われ，gcc等のtoolchain[56)]や，標準ライブラリ等が作られてきた。これらは多くのアーキテクチャに対してクロス開発を行うことのできるもので，OS等の開発には基盤ツールとして重要な役割を果たす。

1985年から始まり1987年に世に出たgccのリリース歴[57)]で特筆しておくべきは，gcc2.5の時代が安定しておりほぼバグも無かった。gcc2.7ではstrength reductionコードにバグが混入してi386のコードに影響したため，最適化フラグ-O2には-fno-strength-reduceを組み合わせて使えというbad knowhowが伝わった。次のgcc2.8はC++98の取込を狙って作られたが，cygnusがpentiumで

53）例えばwikipediaのhttps://ja.wikipedia.org/wiki/Linuxディストリビューションの比較
54）http://itpro.nikkeibp.co.jp/article/COLUMN/20100729/350784/
55）https://www.gnu.org/
56）http://www.fsij.org/monthly-meetings/2011/gnu-toolchain.htmlが判りやすい説明である
57）https://gcc.gnu.org/releases.html

のより速いコード生成に特化したegcsをforkさせ，結局egcs側がGCC Steering Committeeとして開発の主導を取る形で統合されてgcc2.95となった。ここにもまだバグがあり，C99規格の取込も不完全であった。

ともあれ，Linuxの誕生で自由なソフトウェアだけで自由なるOSの完成は一応実現した。

1992年のLinux V0.12からGPL[58]をライセンスとして採用している。現在はGPLv2である。

LinusがGPLを選択した理由は，「コードを第三者に自由に提供できるようにすること」と「改変されたコードも同じように第三者に自由に提供できるようにすること」の出来るライセンスだからと述べ，更には，「問題の適切な解決法が見つかったときに，分岐を再度統合できる」ことも挙げている。

また，Linux特有のカーネル開発モデルは，Eric Raymondの著書「伽藍とバザール」[59]の中でバザール方式と呼ばれるようになった。

GPLは個人が使う分にはあまり問題とならないが，企業が商品開発用のシステムとして使うには使い辛いライセンスであり，特に知的所有権やNDAが関係してソースコードを公開出来ない分野（組み込み機器等）で問題になることが多い。実際にはアプライアンス等で多く使われているが，GPLに従ってソースコードが開示できるのかが疑問なものもある。

GPLの排除について少しだけ述べると，GNUのコンパイラであるGCCがGCC4.2.2以降GPLv3というライセンスに変更された。このライセンスを忌避するFreeBSD側はGCC4.2.1を長く使い続けていた。AppleはObjective-Cを愛用していたが，GCCがこれに注力せずGo言語に手を出して，勝手にライセンスをGPLv3に変更したので，Appleが激怒してClang/LLVMへの支援を強めてデフォルトコンパイラーをClang/LLVMに置き換えた。FreeBSDもFreeBSD10.0からClang/LLVMへの置き換えを行った。

58）例えばhttps://ja.wikipedia.org/wiki/GNU_General_Public_License参照
59）http://cruel.org/freeware/cathedral.html，原文も訳もcopyleft.

Linuxと BSD系では，ユーザコミュニティの違いがあるように思われる。1997年，筆者が後に日本リナックス協会の初代会長となる生越昌己氏とお会いしたときに尋ねると，Linuxは初心者大歓迎，初心者に優しくしようがモットーでユーザ数を集めるのが第一の目標と言っていた記憶がある。初心者の訳のわからない質問に根気よく答えていく姿勢がコミュニティにあった。また，バイナリでパッケージ類をインストールする文化が早くからあったというもの，当時は文化的に異なる象徴の一つであった。他には，Anaconda（アナコンダ）[60]といったRed HatやFedora CoreのGUI環境のOSのインストーラの格好良さ，分かり易さが人気の一つともなった。

それに比較すると，BSD側のコミュニティは判るやつだけついてこい，RTFM[61]的な，初心者に優しく無い雰囲気があった。筆者はBSD側であった。

どのOS（BSD）を選択するか

これまでも，ネットワークプログラミング演習はUNIX環境で行ってきており，今回もUNIX環境を仮想計算機に導入する。これまでの環境は，過去の本学情報処理センターの機種更新について2世代を振り返ると，古い順に富士通PRIMEPOWER 450（XeonMP3.0GHz 4CPU, 8GB Memory）にSunのSolaris9とIBM BladecenterHS22（Xeon E5540(4)(2.53GHz), 8GB Memory） にRed Hat Linux 5であった。Solaris9はUNIX System V系の流れを組むOSであり，ネットワーク・スタックはSTREAMという概念で作られている。

ネットワークプログラミングを行う場合，利用者から見たOSの違いに関しては，一般的なソケットを使ったTCPやUDPを使ったプログラムであれば，あまり大きな差はないと思われる。目に付く差異はLinuxではstruct sockaddr_inのメンバーにsin_lenがない点である。

60）https://ja.wikipedia.org/wiki/Anaconda
61）Read The F***ing Manualの略

しかし，rawソケットを用いたプログラムとなると，ネットワークスタックの構成が全面に出てくるのでOS毎に対応する記述を行う必要がある。そのため，実習用のテキストでは，SunOSのnit，BSD系のbpf，System VのSTREAM，そしてLinux系についての説明を行っている。

さて，今回の実習用環境の構築においては，仮想環境下での使用となるので，GUI環境は特に必要としない。Linuxに関しては，前回の実習ではセンターの共用サーバに商用版のRed Hat Linuxが入れられており，アップデート等は業者なりセンター側で行われるので，PCからsshでログインして使用するだけであればほぼ問題はなかった。しかし，自分で選択して環境を構築するとなると，どのディストリビューションを選択するか，またどれを選択しても大概GUI環境である点，GPLを避けたい点等を考慮するとLinuxは選択肢からは外れる。Arch Linux[62)]のようなシンプルなものもあるが，そのシンプルさは開発者に向くものである。

TCP/IPのネットワークスタックが4.2BSDから広まったことを考えると，やはりOSはBSD系を選択することとなる。そこでどのBSD系を選択するかを考えると，まずGUIが不要であるので，PC-BSDやGhostBSDは外れる。また，ネットワーク実習で色々とネットワークを使ってパケットを送受信することを考えると，セキュリティが厳しいものは面倒が多いのでOpenBSDも外れる。更に，OSの構造がBSD系から変化していったDragonFly BSDはi386サポートが打ち切られており，参考となる本や資料等が少ないので，今回は採用を見送る。HardenedBSDやNextBSDはFreeBSDに近いが使用経験が無いので今回は見送る。

そうなると，候補はFreeBSDとNetBSDというBSDの後継者の２つに絞られる。OS自体のインストールに関しては，両者とも特に大きな違いはない。しかし，OSのバージョンアップや追加する開発環境ソフトウェアの導入に関しての違いがある。FreeBSDはfreebsd-updateを使って，OS自体のアップデー

62）https://www.archlinux.org/

トが可能で，開発環境もpkgを使ったバイナリパッケージでの導入とアップデートが可能である。これに対して，NetBSDでのOSのアップデートはOSのソースコードからのコンパイルとアップデートインストールとなり，開発環境の追加はpkgを用いたソースコードからのコンパイルとなる。

作成した仮想計算機の環境は，学生の手に渡って後は学生が管理していくことを考慮すると，バージョンアップに関しての手間はFreeBSDの方が簡単であることから，今回はFreeBSDを実習用の環境として選択することとした。

また，プログラミングの手助けとなる有益な本も多数出版されており［22］［23］［24］［25］［26］［27］［28］，その日本語訳もある。

環境構築の実際

まず，FreeBSDのどのバージョンで環境を構築するかを決める必要がある。FreeBSDはcurrentとstableの開発ブランチがあり，stableからrelengを経てreleaseが出される。現在のProduction releaseは10.3であり，次期releaseである11.0が現在relengでRCの状況である[63)]。予定では9/9にアナウンスされるのであるが，現状では初物ではいかずに，10.3で行う予定である。kernelはgenericのままで，特に変更はしない。

VMwareで，26GB程度のハードディスクを確保して，メモリを1GBとしたamd64の仮想マシンを作り，そこにFreeBSD10.3をISOイメージからインストールする。ハードディスクはGPTパーティションで使い，10.3のインストーラに付属するzfs rootを使った方式でインストールを行う。標準的なインストールで十分である。ufsと比べてzfsがまだ安定度が低いという声もあるが，今回の演習程度の負荷であれば問題がないであろうという判断からである。実際に10.3をzfsを使ってVMware上で運用しているが，currentのbuild程度の負荷であれば特段問題は出ていない。

63）https://www.freebsd.org/releases/11.0R/schedule.html

一度インストールが済めば，セキュリティアップデート等はfreebsd-updateを用いて行うことが可能であり，アップデートを行っておく。また，現在10.3であるが11.0へのアップデートもfreebsd-updateで行うことが可能である。

次にpkgを導入する。一旦導入すれば，アップデートはpkg updateとpkg upgradeで行える。portsとpkgを同時に使うのが大変なので，今回の実習ではpkgだけで必要なアプリケーションを導入する。デスクトップ環境を使う予定はないので，X11の導入はしない方向で行う。一応windows10側のcygwinでXサーバを動かすことが可能であるから，そのような手法が使いたい学生は自分で入れて使ってもらうこととする。

インストールしておくパッケージは，subversion, git, emacs-nox11, magit, sudo, screen, bash, astyle, libuv, libev, nss_ldap, pam_ldap, open-vm-tools-nox11, socat, ttcp, nc等を考えている。

次に，後学のためを含めて，以下のように freebsd-currentもソースコードを入手して，ビルド可能な環境を残しておく。

```
# zfs create -o mountpoint=/usr/head zfsroot/head
# svn checkout https://svn.FreeBSD.org/base/head /usr/head
```

アカウント管理

本学センターの提供するPCやメールの環境は，LDAPを用いて認証を行っている。このLDAPの情報をそのまま使って，今回の演習環境の認証ができることが望ましい。nss_ldapを使って現在のセンターのLDAPの認証をそのまま利用可能か調査の上，可能であれば利用する。同時に，本学センターでＪドライブと呼んでいる，学生のホームディレクトリが置かれているファイルサーバの当該学生の部分をautomountによりマウントできれば更に良いので，可能性を調査する。

この方法がうまくいかなければ，freebsdというユーザを作って，各自がこのユーザでログインするという方法をとる。人数がそれほど多くないので，

freebsd01からfreebsdNといった人数分のアカウントを作るのも１つの方法ではあるが，他人の環境に入って作業することがないので，ユーザ用に１つアカウントで十分であろう。そして，sudoでrootになることを許可しておく。LDAPを用いることができた場合もsudoでrootになることを可能としておく。これは，仮想環境の管理は学生が行うためである。

その後の調査の結果，本学で使用しているLDAPとの連携が難しいことが判明したため，今回はfreebsdというユーザでの使用とした。

ユーザ名freebsdでの演習となる場合は，最初のログイン時にパスワードを各自変更させる。

なお，rootのパスワードはあらかじめ適当に決めて設定しておく。sudoで学生があとで変更が可能である。rootについては，authorized_keysに筆者のsshのキーをあらかじめ登録しておく。そして，/etc/ssh/sshd_configでPermitRoot-Login yesとして管理者として学生の仮想環境に入れるように準備をしておく。演習終了後も仮想環境を使ってプログラミングをする場合は，authorized_keysから筆者のキーを消せばよい。同時に，/etc/ssh/sshd_configでPasswordAuthentication yesも設定して，sshのパスワード認証で使用できるようにしておく。

１台のサーバ上で全員が演習をする場合はそのサーバにログインするだけで全員の状況が管理できたが，各々がVMを占有すると個別の管理は手間がかかる。

学生は自身が使用するPC上のVMwareでこのFreeBSD環境を起動し，コンソールの上でscreenを使って，もしくはPC上のターミナルソフト（teraterm）を使ってFreeBSD環境にログインして演習を行うことになる。また，最低限の環境の保守も自身がsudoでrootとなって行うこととなる。

システム管理

/etc/rc.conf等，システムが立ち上がる時点でどのような設定をするかを決めておく必要がある。まず，ネットワークに関してはVMwareの設定でネット

ワークアダプタはシステムのEthernetに追加という形で設定し，FreeBSD側ではDHCPでIPv4アドレスを取得するようにする。本学の情報処理センターの学生用PCはIPv4に関してはプライベートアドレスを使用している。次に，本演習ではIPv6も使用しているのでIPv6のアドレスを設定する必要がある。本来であれば，本学に割り当てられたグローバルなアドレスを使って設定ができると良いのであるが，IPv6 Router Advertisementを行ってPC側がIPv6のグローバルアドレスを取得されるのを嫌って，現状ではルータによって設定がなされていないので，Link LocalなIPv6アドレスの設定のみで行うことにする。演習は同一のリンクセグメント上で行うので，他の学生の環境にLink LocalなIPv6アドレスで接続にいくことが可能である。

次に，最近のUNIX環境ではあまり動作させることがなくなったinetdを，プライベートアドレスな環境で外部からの攻撃に晒されないという点を生かして，動作させることとする。inetd_enable=“YES”を/etc/rc.confに追加する。/etc/inetd.confで はftp, telnet, daytime, time, echo, discard, chargenをIPv4, IPv6両方で開けておく。

さらに，システムの時刻を正しく保っておくために，ntpdを起動する。本学ではntpサーバを独自に立ち上げているので，ntp.confで学内のntpサーバのみを指定することとし，システムの起動時にntpdateによって時刻を同期する。

以上をまとめた/etc/rc.confは以下の通りとなる。

```
hostname=""
keymap="jp.106.kbd"

ifconfig_em0="DHCP"
ipv6_activate_all_interfaces="YES"

vmware_guest_vmblock_enable="YES"
vmware_guest_vmhgfs_enable="YES"
vmware_guest_vmmemctl_enable="YES"
vmware_guest_vmxnet_enable="YES"
```

```
vmware_guestd_enable="YES"
sshd_enable="YES"
# Set dumpdev to "AUTO" to enable crash dumps, "NO" to disable
dumpdev="AUTO"
zfs_enable="YES"
tcsd_enable="YES"
nscd_enable="YES"
inetd_enable="YES"  # Run the network daemon dispatcher (YES/NO).
ntpdate_enable="YES" # Run ntpdate to sync time on boot (or NO).
ntpd_enable="YES"
```

演習を終えて

実際の演習をこの環境で行った。幾つかの問題が発生したが，概ね問題なく演習を終えることができた。

生じた問題としては，学生が USB上のFreeBSDに最初にログインをして，パスワードを変更した後，そのパスワードを忘れてしまうという事象が2件発生した。筆者が管理者としてネットワーク経由で入り，パスワードを変更することで問題なく演習が続けられた。外部から管理者としてシステムに入れるためのsshのauthorized_keysの登録はこのような環境では役に立つ。

更には，VMwareでゲストOSを立ち上げると，ゲストOSの実行に管理者権限を要求されるという事象も発生した。演習最終日まで原因が不明であったため，かなりの回数が発生した。一旦この状況になると，管理者権限を持たないためゲストOSの立ち上げができない状況が続き，学生がPCを移動して回避していた。

原因は，Windows側でPCのブート時にUAC（ユーザーアカウント制御）が正常に立ち上がる前にVMwareを立ち上げると，VMware Authorization Serviceが正常に起動せず，問題が生じることが判明した。本学のドメインコントローラでポリシーを変更して対応することとした。

USB上の実際のFreeBSD 10.3のイメージの大きさは，OSのソースコードや

パッケージを幾つか入れた状態で6GB程度であった。今回実習を行った本学情報処理センター第４実習室のネットワーク環境が100Mbpsであったため，ネットワークディスク上にイメージを置くことを断念したが，実際どの程度時間がかかるかを試してみた。まずUSBのイメージを第４実習室でネットワークドライブにコピーするために要した時間は６－７分程度であった。これは１人だけで行ったので同時に数十人が行うとこの時間では無理であろう。また，ネットワークドライブからVMwareのゲストOSとして立ち上げる時間はUSBからと大きな差はみられなかった。これも１人だけで行った状況であるため，演習のように同時に数十人が行うと厳しいと思われる。但し，今回は1Gbpsのネットワークで接続された他の実習室からであれば無理なく行える可能性はある。

学生は演習の後，10月末までにレポートを提出することとなっているので，完成に至らなかった課題を自習したりレポート作成のために，実習で使用したUSBメモリを10月末まで貸与している。OSのイメージを学生のネットワークドライブにコピーして使うことも可能である。また，作成したプログラムのファイルをレポート作成のためにPC側に持ってくるためには，Windows 10のpowershell上でftpを使うのが簡便である。PC側にscp，sftp等のクライアントが入っていればそれを使うことも可能である。

今回，必要に迫られてネットワークプログラミング演習のための環境を構築し，成功裏に演習を終わらせることができた。環境構築に際して，UNIXの系譜を辿り，現状のBSD系のOS環境を概観し，今回の環境としてVMware上のFreeBSD 10.3を選択した。次年度以降も同様の演習を行うので，少なくとも次回の本学センターの機種更新までは同様の方式での実施となるであろう。

参考文献

［１］Salus, Peter H., A Quarter Century of UNIX, ACM Press/Addison-Wesley Publishing Co., 1994.

［２］Don Libes and Sandy Ressler, Life with UNIX: A Guide for Everyone, Prentice Hall, 1989.

［３］藤田昭人，Unix考古学Truth of the Legend, KADOKAWA, 2016.

［４］Ritchie, Dennis M. and Thompson, Ken, The UNIX Time-sharing System, Commun. ACM, Vol.17, No.7, pp.365-375, Jul. 1974.

［５］John Lions, Lions' Commentary on Unix 6th Edition, Peer to Peer Communications, 1977.

［６］Andrew S. Tannenbaum, Operating Systems: Design and Implementation (Prentice-Hall Software Series), Prentice Hall, 1987.

［７］UNIX Time-Sharing System, Bell System Technical Journal, Vol.57, No.6 Part 2, American Telephone and Telegraph Company, Murray Hill, N.J., July-August 1978.

［８］The UNIX System, AT&T Bell Laboratories Technical Journal, Vol.63, No.8, American Telephone and Telegraph Company, Short Hills, N.J., October 1984.

［９］パソコンUNIX：[PC-9801] PC-UXで知るUNIXの世界，アスキー書籍編集部編著，アスキー出版局，1985.

[10] Maurice J. Bach, The Design of the UNIX Operating System, Prentice Hall,1986.

[11] Berny Goodheart and James Cox, The Magic Garden Explained: The Internals of UNIX System V Release 4 an Open Systems Design, Prentice Hall, 1994.

[12] Samuel J. Leffler, Marshall Kirk McKusick, Michael J. Karels and John S. Quarterman, The Design and Implementation of the 4.3 BSD UNIX Operating System, Addison-Wesley, 1989.

[13] Marshall Kirk McKusick, Keith Bostic, Michael J. Karels and John S. Quarterman, The Design and Implementation of the 4.4 BSD Operating System, Addison-Wesley, 1996.

[14] Marshall Kirk McKusick and George V. Neville-Neil, The Design and Implementation of the FreeBSD Operating System, Addison-Wesley Professional, 2004.

[15] Marshall Kirk McKusick, George V. Neville-Neil and Robert N.M. Watson, The Design and Implementation of the FreeBSD Operating System (2nd Edition), Addison-Wesley Professional, 2014.

[16] Vinton G. Cerf and Robert E. Khan, A Protocol for Packet Network

Intercommunication, IEEE TRANSACTIONS ON COMMUNICATIONS, Vol.22, No.5, pp.637–648, May 1974.
[17] J. Postel, Assigned numbers, RFC750, IETF, Sep. 1978. url=http://www.ietf.org/rfc/rfc750.txt
[18] J. Postel, Transmission Control Protocol, RFC 793, IETF, Sep. 1981. url=http://www.ietf.org/rfc/rfc793.txt
[19] J. Postel, Internet Protocol, RFC 791, IETF, Sep. 1981. url=http://www.ietf.org/rfc/rfc791.txt
[20] J. Postel, NCP/TCP transition plan, RFC 801, IETF, Nov. 1981. url=http://www.ietf.org/rfc/rfc801.txt
[21] Robert F. Gurwitz, VAX-UNIX Networking Support Project Implementation Description, IEN 168, Jan. 1981. url=http://www.postel.org/ien/txt/ien168.txt
[22] Kevin R. Fall and W. Richard Stevens, TCP/IP Illustrated, Volume 1: The Protocols (2nd Edition) (Addison-Wesley Professional Computing Series), Addison-Wesley Professional, 2011.
[23] Gary R. Wright and W. Richard Stevens, TCP/IP Illustrated: The Implementation, Vol. 2, Addison-Wesley Professional, 1995.
[24] W. Richard Stevens, Bill Fenner and Andrew M. Rudoff, Unix Network Programming, Volume 1: The Sockets Networking API (3rd Edition), Addison-Wesley Professional, 2003.
[25] W. Richard Stevens, UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition, Prentice Hall, 1998.
[26] W. Richard Stevens and Stephen A. Rago, Advanced Programming in the UNIX Environment, 3rd Edition, Addison-Wesley Professional, 2013.
[27] Eric S. Raymond, The Art of UNIX Programming (The Addison-Wesley Professional Computing Series), Addison-Wesley, 2003.
[28] Bruce Molay, Understanding UNIX/LINUX Programming: A Guide to Theory and Practice, Pearson, 2002.
[29] R. N. M. Watson and J. Woodruff and P. G. Neumann and S. W. Moore and J. Anderson and D. Chisnall and N. Dave and B. Davis and K. Gudka and B. Laurie and S. J. Murdoch and R. Norton and M. Roe and S. Son and M. Vadera, CHERI: A Hybrid Capability-System Architecture for Scalable Software Compartmentalization, 2015 IEEE Symposium on Security and Privacy, pp.20–37, May, 2015.